Project-01-Identifying-predictors

July 3, 2022

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
%matplotlib inline
```

Load the dataset

[3]:
```
df = pd.read_csv(r'C:\Users\krzys\Desktop\automobileEDA.csv')
```

Check the dataset

[4]:
```
df.dtypes
```

[4]:
```
symboling              int64
normalized-losses      int64
make                  object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
```

```
highway-mpg            int64
price                float64
city-L/100km         float64
horsepower-binned     object
diesel                 int64
gas                    int64
dtype: object
```

[5]:

```python
#Missing values
missing_data = df.isnull()

for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")

#Fix missing values in stroke column
df.interpolate(inplace=True)  #fix missing in stroke
df.drop(['horsepower-binned'], inplace=True, axis=1)
```

```
symboling
False    201
Name: symboling, dtype: int64

normalized-losses
False    201
Name: normalized-losses, dtype: int64

make
False    201
Name: make, dtype: int64

aspiration
False    201
Name: aspiration, dtype: int64

num-of-doors
False    201
Name: num-of-doors, dtype: int64

body-style
False    201
Name: body-style, dtype: int64

drive-wheels
False    201
Name: drive-wheels, dtype: int64
```

```
engine-location
False    201
Name: engine-location, dtype: int64

wheel-base
False    201
Name: wheel-base, dtype: int64

length
False    201
Name: length, dtype: int64

width
False    201
Name: width, dtype: int64

height
False    201
Name: height, dtype: int64

curb-weight
False    201
Name: curb-weight, dtype: int64

engine-type
False    201
Name: engine-type, dtype: int64

num-of-cylinders
False    201
Name: num-of-cylinders, dtype: int64

engine-size
False    201
Name: engine-size, dtype: int64

fuel-system
False    201
Name: fuel-system, dtype: int64

bore
False    201
Name: bore, dtype: int64

stroke
False    197
True       4
```

```
Name: stroke, dtype: int64

compression-ratio
False    201
Name: compression-ratio, dtype: int64

horsepower
False    201
Name: horsepower, dtype: int64

peak-rpm
False    201
Name: peak-rpm, dtype: int64

city-mpg
False    201
Name: city-mpg, dtype: int64

highway-mpg
False    201
Name: highway-mpg, dtype: int64

price
False    201
Name: price, dtype: int64

city-L/100km
False    201
Name: city-L/100km, dtype: int64

horsepower-binned
False    200
True       1
Name: horsepower-binned, dtype: int64

diesel
False    201
Name: diesel, dtype: int64

gas
False    201
Name: gas, dtype: int64
```

Correlations

[6]:
```
#Quick look
df_corr = df.corr()
```

```python
df_corr.sort_values(by=['price'])
```

[6]:

	symboling	normalized-losses	wheel-base	length
highway-mpg	0.036233	-0.181877	-0.543304	-0.698142
city-mpg	-0.035527	-0.225016	-0.470606	-0.665192
gas	0.196735	0.101546	-0.307237	-0.211187
peak-rpm	0.279740	0.239543	-0.360305	-0.285970
symboling	1.000000	0.466264	-0.535987	-0.365404
compression-ratio	-0.182196	-0.114713	0.250313	0.159733
stroke	-0.006564	0.055836	0.157438	0.123525
diesel	-0.196735	-0.101546	0.307237	0.211187
normalized-losses	0.466264	1.000000	-0.056661	0.019424
height	-0.550160	-0.373737	0.590742	0.492063
bore	-0.140019	-0.029862	0.493244	0.608971
wheel-base	-0.535987	-0.056661	1.000000	0.876024
length	-0.365404	0.019424	0.876024	1.000000
width	-0.242423	0.086802	0.814507	0.857170
city-L/100km	0.066171	0.238567	0.476153	0.657373
horsepower	0.075819	0.217299	0.371147	0.579821
curb-weight	-0.233118	0.099404	0.782097	0.880665
engine-size	-0.110581	0.112360	0.572027	0.685025
price	-0.082391	0.133999	0.584642	0.690628

	width	height	curb-weight	engine-size	bore
highway-mpg	-0.680635	-0.104812	-0.794889	-0.679571	-0.591309
city-mpg	-0.633531	-0.049800	-0.749543	-0.650546	-0.582027
gas	-0.244356	-0.281578	-0.221046	-0.070779	-0.054458
peak-rpm	-0.245800	-0.309974	-0.279361	-0.256733	-0.267392
symboling	-0.242423	-0.550160	-0.233118	-0.110581	-0.140019
compression-ratio	0.189867	0.259737	0.156433	0.028889	0.001263
stroke	0.188681	-0.062214	0.167397	0.204933	-0.055376
diesel	0.244356	0.281578	0.221046	0.070779	0.054458
normalized-losses	0.086802	-0.373737	0.099404	0.112360	-0.029862
height	0.306002	1.000000	0.307581	0.074694	0.180449
bore	0.544885	0.180449	0.644060	0.572609	1.000000
wheel-base	0.814507	0.590742	0.782097	0.572027	0.493244
length	0.857170	0.492063	0.880665	0.685025	0.608971
width	1.000000	0.306002	0.866201	0.729436	0.544885
city-L/100km	0.673363	0.003811	0.785353	0.745059	0.554610
horsepower	0.615077	-0.087027	0.757976	0.822676	0.566936
curb-weight	0.866201	0.307581	1.000000	0.849072	0.644060
engine-size	0.729436	0.074694	0.849072	1.000000	0.572609
price	0.751265	0.135486	0.834415	0.872335	0.543155

	stroke	compression-ratio	horsepower	peak-rpm
highway-mpg	-0.035665	0.268465	-0.804575	-0.058598
city-mpg	-0.035333	0.331425	-0.822214	-0.115413

gas	-0.240684	-0.985231	0.169053	0.475812
peak-rpm	-0.061840	-0.435780	0.107885	1.000000
symboling	-0.006564	-0.182196	0.075819	0.279740
compression-ratio	0.187638	1.000000	-0.214514	-0.435780
stroke	1.000000	0.187638	0.099424	-0.061840
diesel	0.240684	0.985231	-0.169053	-0.475812
normalized-losses	0.055836	-0.114713	0.217299	0.239543
height	-0.062214	0.259737	-0.087027	-0.309974
bore	-0.055376	0.001263	0.566936	-0.267392
wheel-base	0.157438	0.250313	0.371147	-0.360305
length	0.123525	0.159733	0.579821	-0.285970
width	0.188681	0.189867	0.615077	-0.245800
city-L/100km	0.038001	-0.299372	0.889488	0.115830
horsepower	0.099424	-0.214514	1.000000	0.107885
curb-weight	0.167397	0.156433	0.757976	-0.279361
engine-size	0.204933	0.028889	0.822676	-0.256733
price	0.082982	0.071107	0.809575	-0.101616

	city-mpg	highway-mpg	price	city-L/100km	diesel \
highway-mpg	0.972044	1.000000	-0.704692	-0.930028	0.198690
city-mpg	1.000000	0.972044	-0.686571	-0.949713	0.265676
gas	-0.265676	-0.198690	-0.110326	0.241282	-1.000000
peak-rpm	-0.115413	-0.058598	-0.101616	0.115830	-0.475812
symboling	-0.035527	0.036233	-0.082391	0.066171	-0.196735
compression-ratio	0.331425	0.268465	0.071107	-0.299372	0.985231
stroke	-0.035333	-0.035665	0.082982	0.038001	0.240684
diesel	0.265676	0.198690	0.110326	-0.241282	1.000000
normalized-losses	-0.225016	-0.181877	0.133999	0.238567	-0.101546
height	-0.049800	-0.104812	0.135486	0.003811	0.281578
bore	-0.582027	-0.591309	0.543155	0.554610	0.054458
wheel-base	-0.470606	-0.543304	0.584642	0.476153	0.307237
length	-0.665192	-0.698142	0.690628	0.657373	0.211187
width	-0.633531	-0.680635	0.751265	0.673363	0.244356
city-L/100km	-0.949713	-0.930028	0.789898	1.000000	-0.241282
horsepower	-0.822214	-0.804575	0.809575	0.889488	-0.169053
curb-weight	-0.749543	-0.794889	0.834415	0.785353	0.221046
engine-size	-0.650546	-0.679571	0.872335	0.745059	0.070779
price	-0.686571	-0.704692	1.000000	0.789898	0.110326

	gas
highway-mpg	-0.198690
city-mpg	-0.265676
gas	1.000000
peak-rpm	0.475812
symboling	0.196735
compression-ratio	-0.985231
stroke	-0.240684

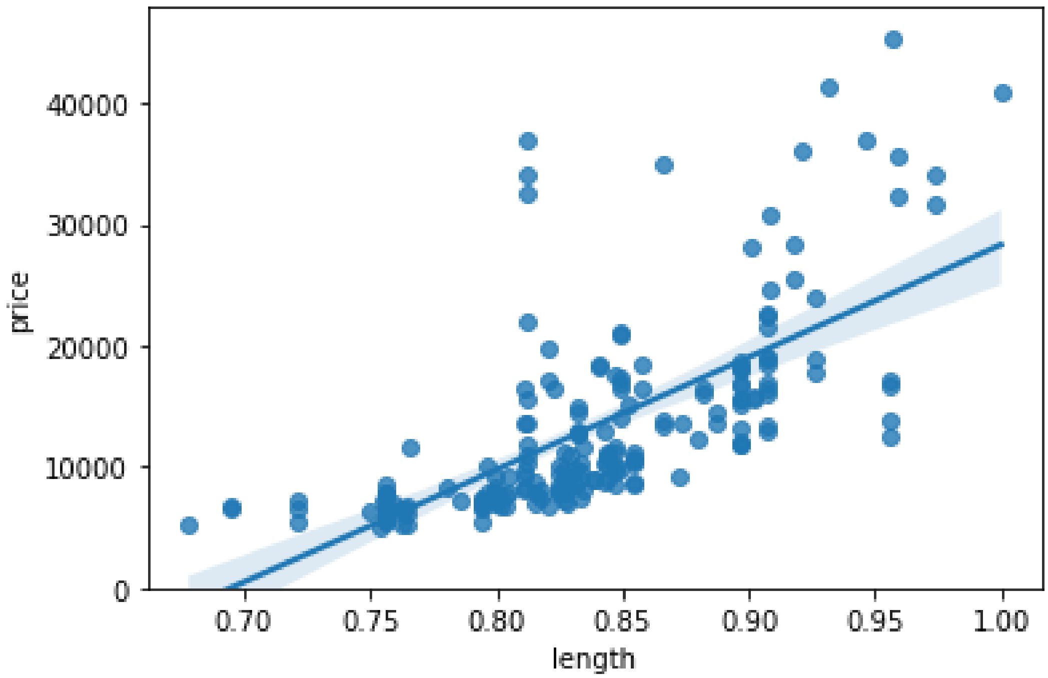
```
diesel              -1.000000
normalized-losses    0.101546
height              -0.281578
bore                -0.054458
wheel-base          -0.307237
length              -0.211187
width               -0.244356
city-L/100km         0.241282
horsepower           0.169053
curb-weight         -0.221046
engine-size         -0.070779
price               -0.110326
```

[7]:
```
#Only the following numerical variables should be considered in further␣
↪analysis (due to moderate or strong
#positive/negative correlation): length, width, curb-weight, engine-size,␣
↪horsepower, city-mpg, highway-mpg,
#wheel-base, bore.
```

[8]:
```
#Detailed Pearson correlation analysis

print('Length')
pearson_coef, p_value = stats.pearsonr(df['length'], df['price'])
print("Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between length and price is␣
↪statistically
#significant, and the linear relationship is moderately strong (~0.691).

sns.regplot(x="length", y="price", data=df)
plt.ylim(0,)
```

```
Length
Correlation Coefficient: 0.690628380448364  P-value: 8.016477466158986e-30
```

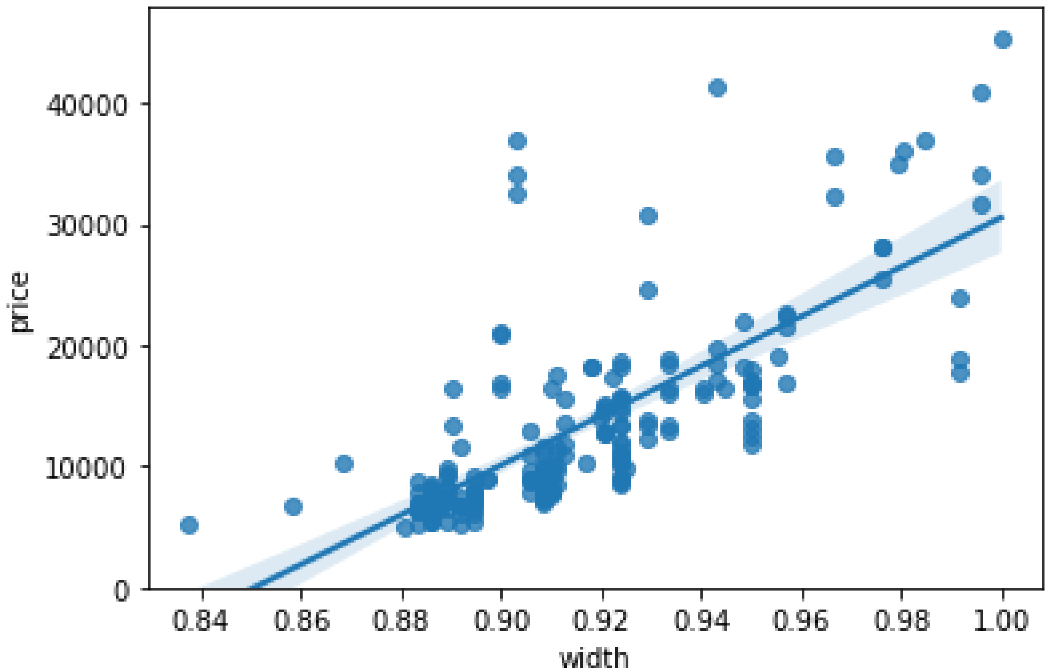
[8]:
```
(0.0, 47867.09963559985)
```



[9]:
```
print('Width')
pearson_coef, p_value = stats.pearsonr(df['width'], df['price'])
print("Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between width and price is␣
 ↪statistically
#significant, and the linear relationship is quite strong (~0.751).

sns.regplot(x="width", y="price", data=df)
plt.ylim(0,)
```

```
Width
Correlation Coefficient: 0.7512653440522674  P-value: 9.200335510481516e-38
```

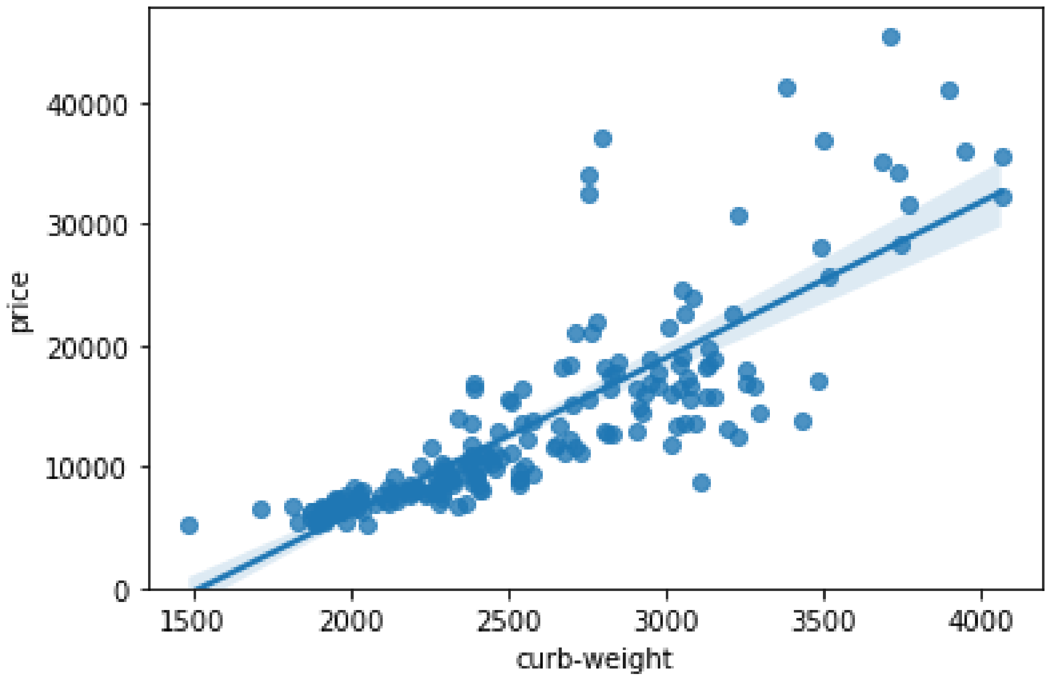
[9]:
```
(0.0, 47914.503239972415)
```



[10]:
```
print('Curb-weight')
pearson_coef, p_value = stats.pearsonr(df['curb-weight'], df['price'])
print( "Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between curb-weight and price is␣
↪statistically
#significant, and the linear relationship is quite strong (~0.834).

sns.regplot(x="curb-weight", y="price", data=df)
plt.ylim(0,)
```

```
Curb-weight
Correlation Coefficient: 0.8344145257702846  P-value: 2.1895772388936914e-53
```

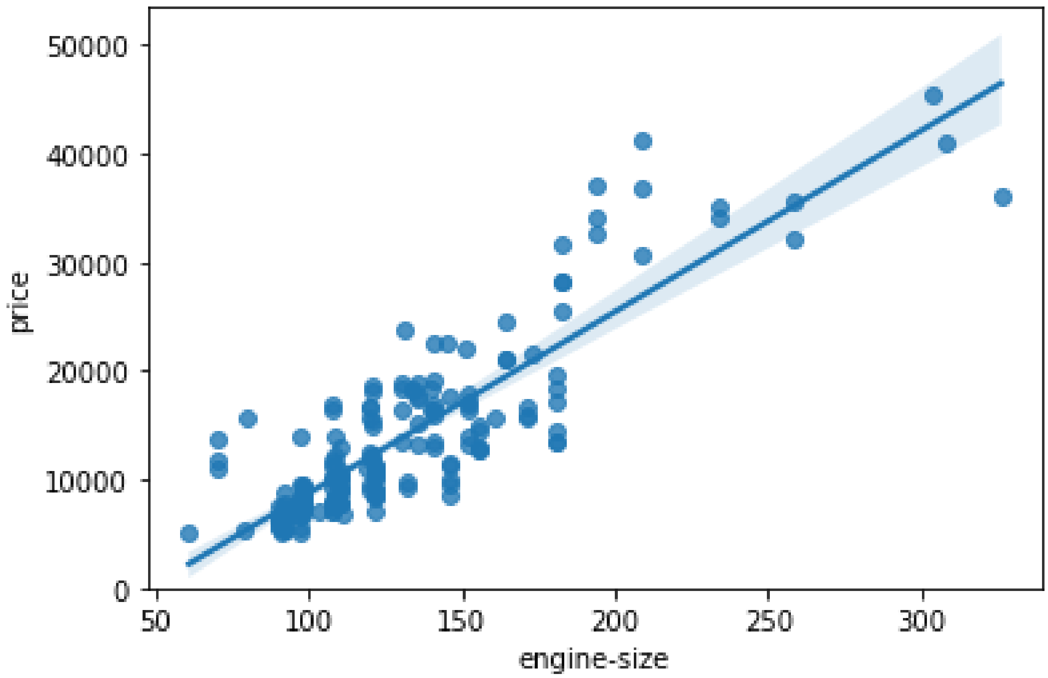
[10]:
```
(0.0, 47754.78706739099)
```



[11]:
```
print('Engine-size')
pearson_coef, p_value = stats.pearsonr(df['engine-size'], df['price'])
print("Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between engine-size and price is␣
↪statistically
#significant, and the linear relationship is very strong (~0.872).

sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```

```
Engine-size
Correlation Coefficient: 0.8723351674455185  P-value: 9.265491622198389e-64
```

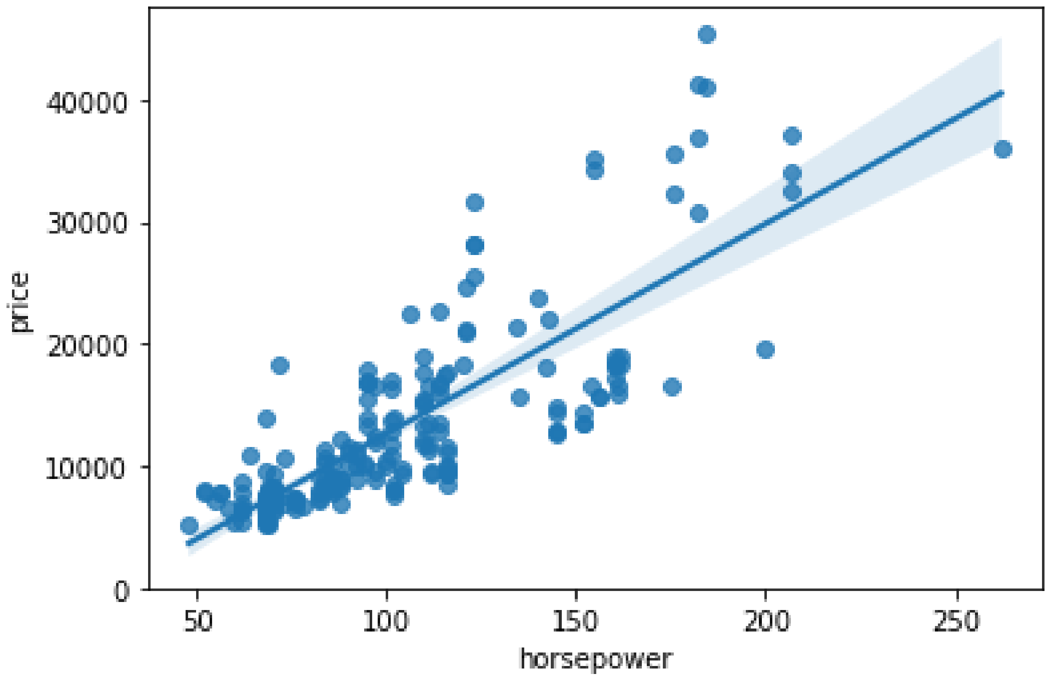
[11]: (0.0, 53414.609178181076)



[12]:
```
print('Horsepower')
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
print("Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between horsepower and price is␣
↪statistically
#significant, and the linear relationship is quite strong (~0.809, close to 1).

sns.regplot(x="horsepower", y="price", data=df)
plt.ylim(0,)
```

```
Horsepower
Correlation Coefficient: 0.809574567003656  P-value: 6.369057428259557e-48
```

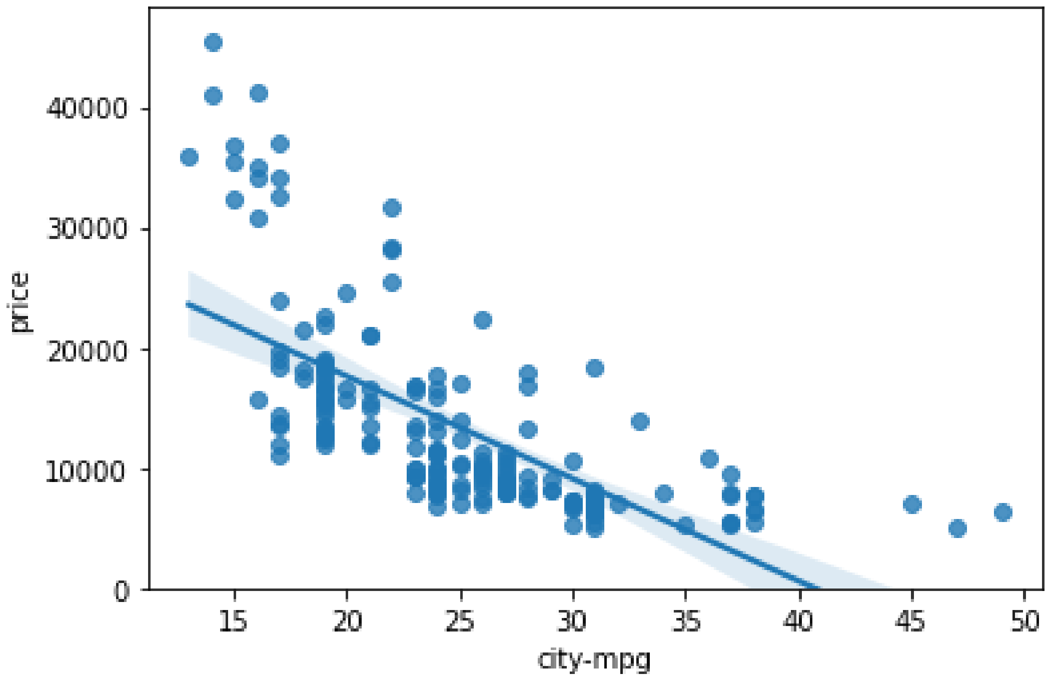
[12]: (0.0, 47540.810858772275)



[13]:
```
print('City-mpg')
pearson_coef, p_value = stats.pearsonr(df['city-mpg'], df['price'])
print("Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between city-mpg and price is␣
 ↪statistically
#significant, and the coefficient of about -0.687 shows that the relationship␣
 ↪is negative and
#moderately strong.

sns.regplot(x="city-mpg", y="price", data=df)
plt.ylim(0,)
```

```
City-mpg
Correlation Coefficient: -0.6865710067844677  P-value: 2.321132065567674e-29
```

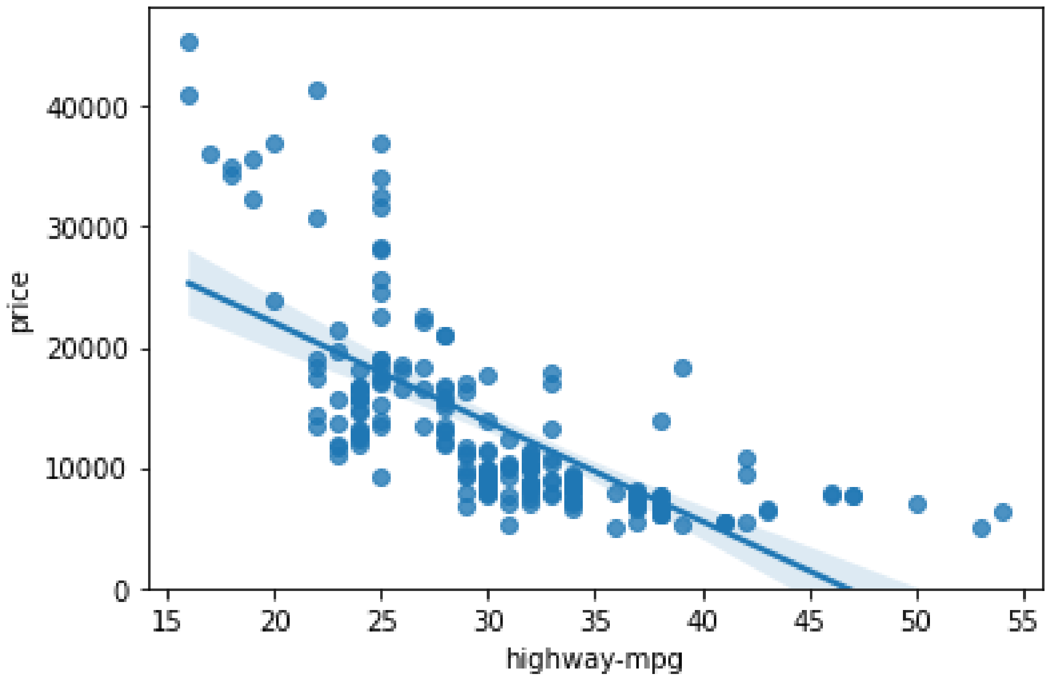
[13]:
```
(0.0, 48246.761410914216)
```




```
[14]: print('Highway-mpg')
pearson_coef, p_value = stats.pearsonr(df['highway-mpg'], df['price'])
print( "Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between highway-mpg and price is
 ↪statistically
#significant, and the coefficient of about -0.705 shows that the relationship
 ↪is negative and
#moderately strong.

sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```

```
Highway-mpg
Correlation Coefficient: -0.7046922650589529  P-value: 1.7495471144477352e-31
```

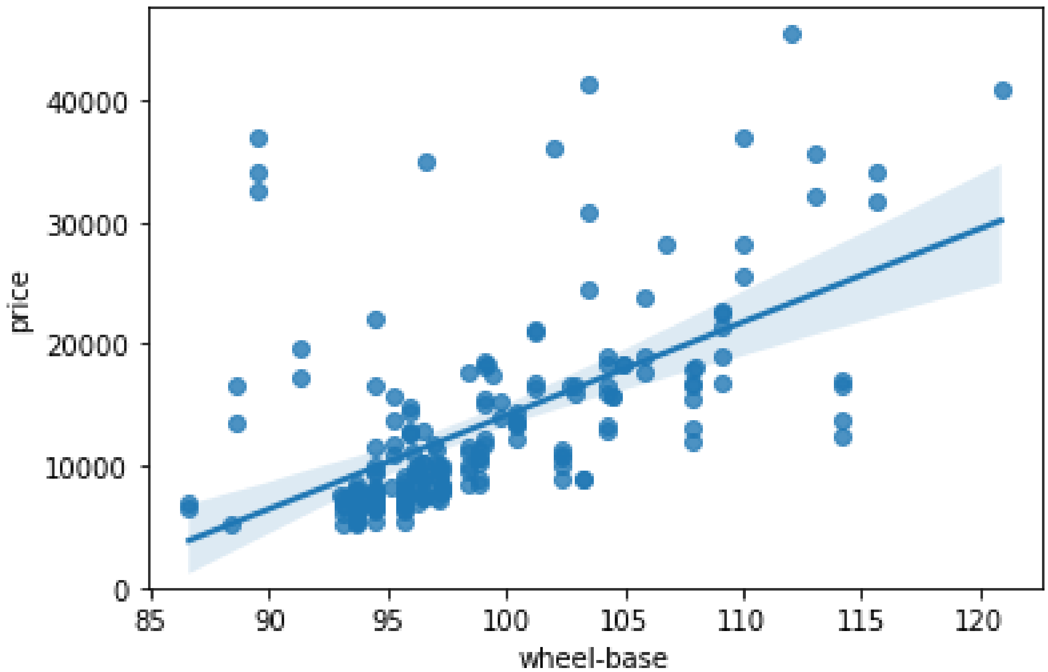
```
[14]: (0.0, 48160.81853259754)
```



[15]:
```
print("Wheel-base")
pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])
print("Correlation Coefficient:", pearson_coef, " P-value:", p_value)
print(" ")
#Since the p-value is < 0.001, the correlation between wheel-base and price is␣
 ↪statistically
#significant, although the linear relationship isn't extremely strong (~0.585).

sns.regplot(x="wheel-base", y="price", data=df)
plt.ylim(0,)
```

```
Wheel-base
Correlation Coefficient: 0.5846418222655081  P-value: 8.076488270732989e-20
```

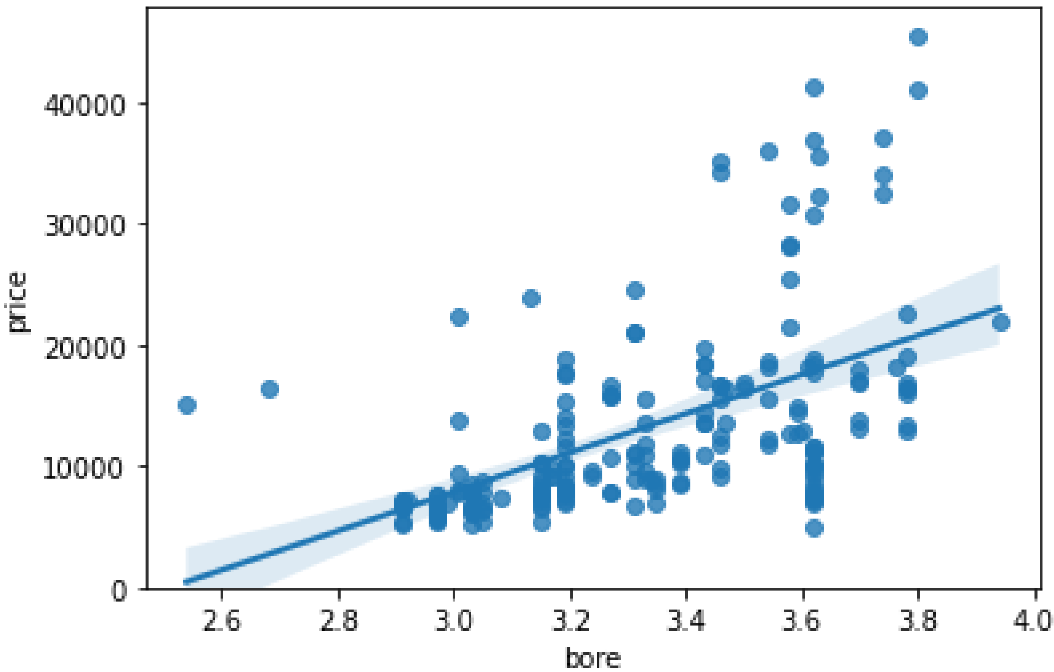
[15]:
```
(0.0, 47614.140319248014)
```



[16]:
```
print('Bore')
pearson_coef, p_value = stats.pearsonr(df['bore'], df['price'])
print("Correlation Coefficient:", pearson_coef, " P-value:", p_value)
#Since the p-value is < 0.001, the correlation between bore and price is␣
↪statistically
#significant, but the linear relationship is only moderate (~0.521)

sns.regplot(x="bore", y="price", data=df)
plt.ylim(0,)
```

```
Bore
Correlation Coefficient: 0.5431553832626602  P-value: 8.049189483935489e-17
```

[16]:
```
(0.0, 47804.113699511036)
```

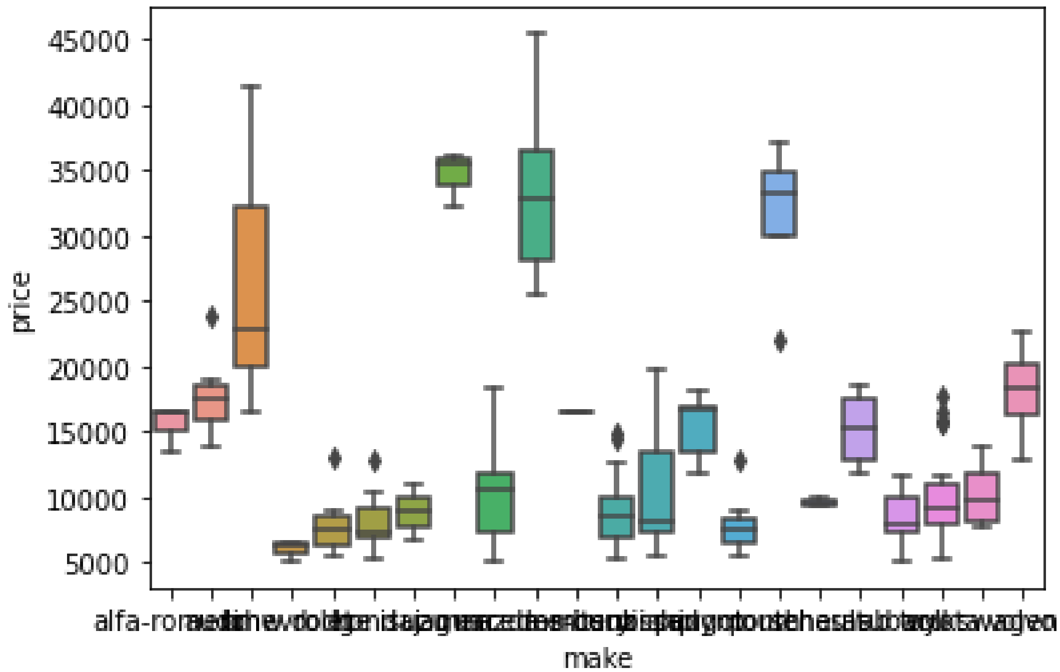


Categorical variables

[18]:
```
#make, aspiration, num-of-doors, body-style, drive-wheels, engine-location,␣
 ↪engine-type, num-of-cylinders, #fuel-system

#Exclude variables that are not appropriate for predictors
```

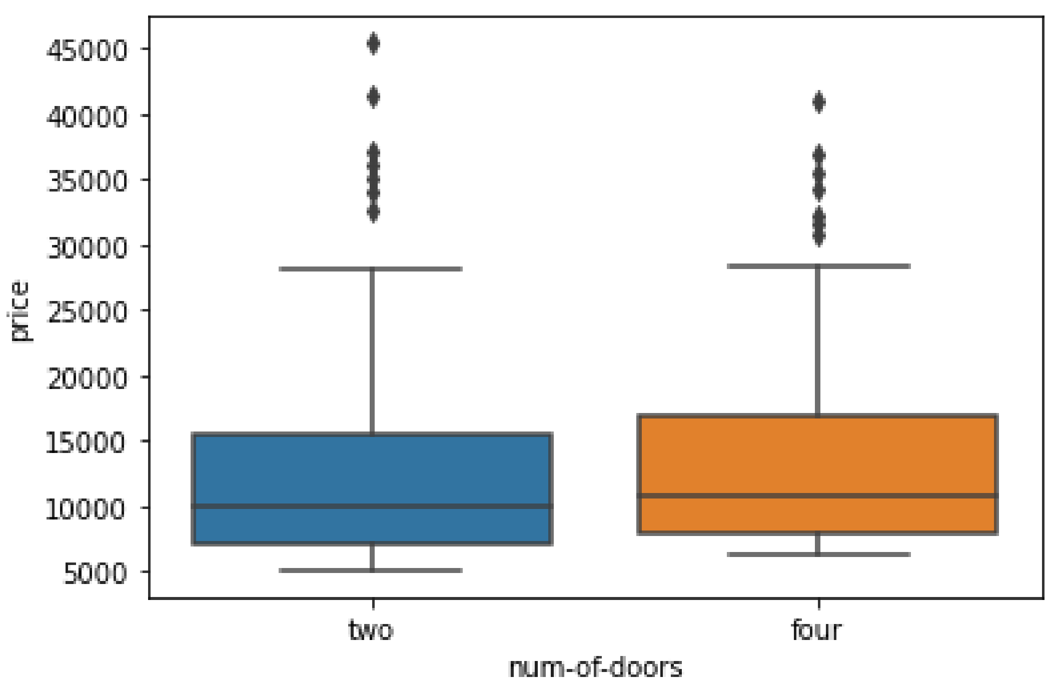
[19]:
```
#Make
sns.boxplot(x="make", y="price", data=df)
#exclude due to overlapping
```

[19]: <AxesSubplot:xlabel='make', ylabel='price'>



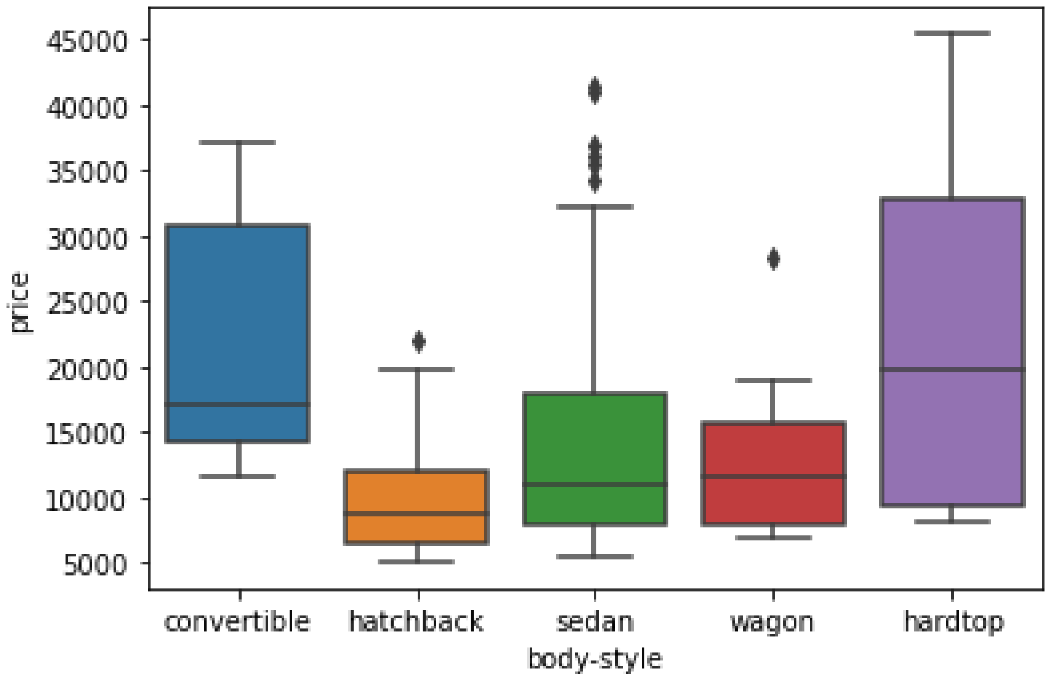
[20]:
```
#Num-of-doors
sns.boxplot(x="num-of-doors", y="price", data=df)
#exclude due to overlapping
```

[20]: <AxesSubplot:xlabel='num-of-doors', ylabel='price'>



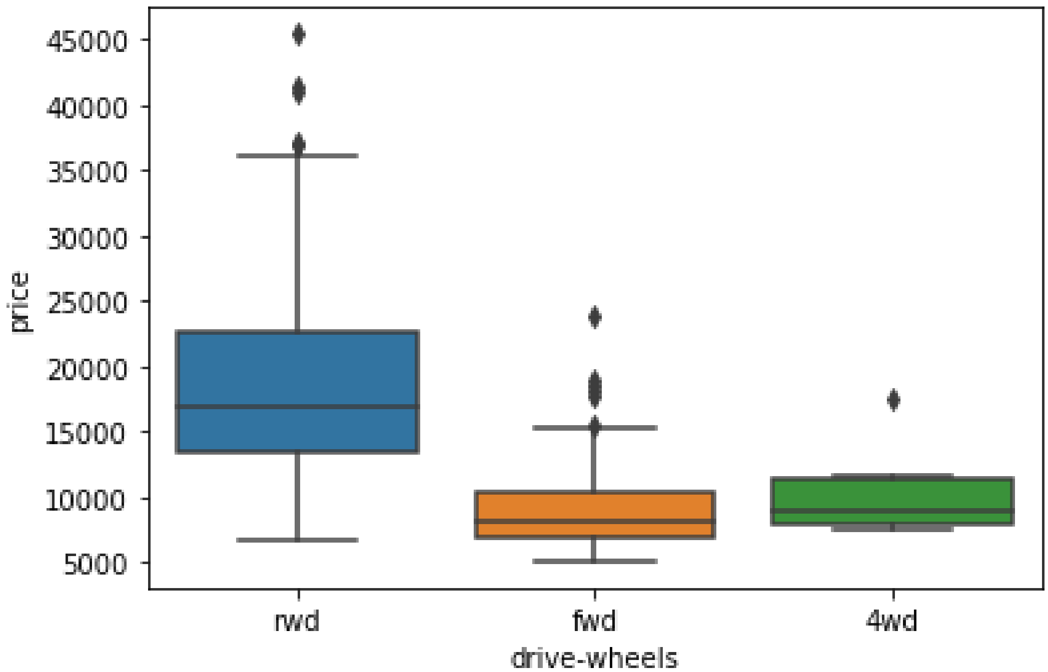
[21]:
```
#Body-style
sns.boxplot(x="body-style", y="price", data=df)
#exclude due to overlapping
```

[21]: <AxesSubplot:xlabel='body-style', ylabel='price'>



[22]:
```
#Drive-wheels
sns.boxplot(x="drive-wheels", y="price", data=df)
```

[22]: <AxesSubplot:xlabel='drive-wheels', ylabel='price'>



[23]:
```
#Count per drive-wheels
drive_wheels_counts = df['drive-wheels'].value_counts().to_frame()
drive_wheels_counts.rename(columns={'drive-wheels': 'value_counts'},␣
↪inplace=True)
drive_wheels_counts.index.name = 'drive-wheels'
drive_wheels_counts
```

[23]:

	value_counts
drive-wheels	
fwd	118
rwd	75
4wd	8

[24]:
```
#Average price for drive-wheels
df_group_one = df[['drive-wheels','body-style','price']]
df_group_one = df_group_one.groupby(['drive-wheels'],as_index=False).mean()
df_group_one.rename(columns={'price': 'avg price'}, inplace=True)
df_group_one
```

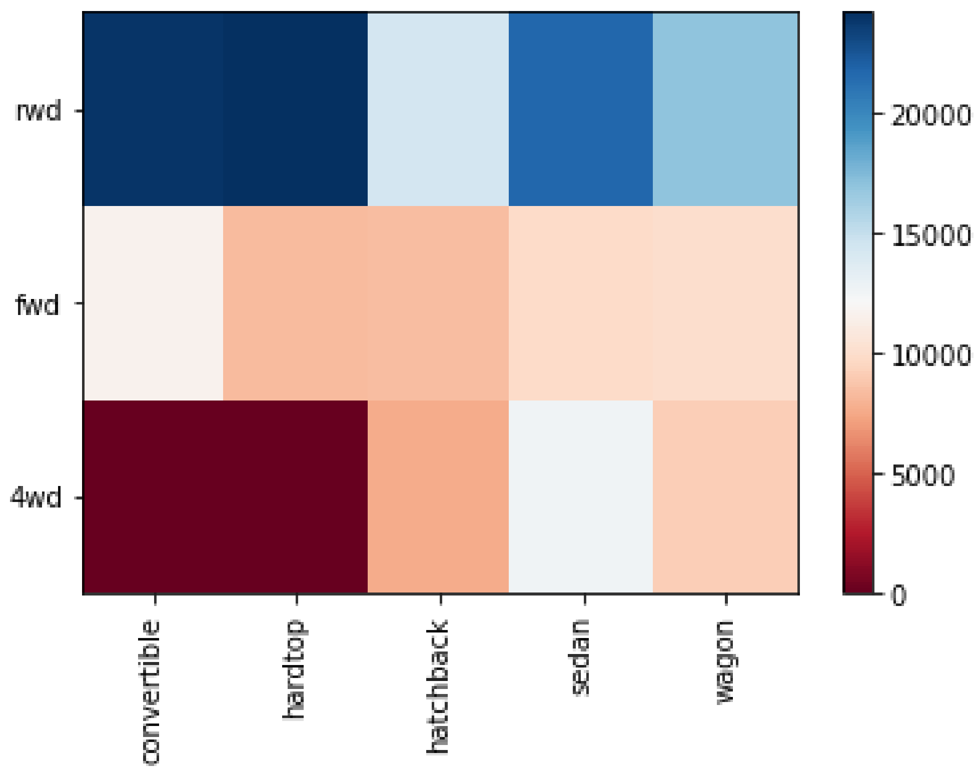
[24]:

	drive-wheels	avg price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

[25]:
```python
#Average price for drive-wheels and body-style
df_gptest = df[['drive-wheels','body-style','price']]
grouped_test1 = df_gptest.groupby(['drive-wheels','body-style'],␣
 ↪as_index=False).mean()
grouped_pivot = grouped_test1.pivot(index='drive-wheels', columns='body-style')
grouped_pivot = grouped_pivot.fillna(0)
grouped_pivot
```

[25]:
```
                   price                                                    \
body-style   convertible       hardtop     hatchback         sedan
drive-wheels
4wd                  0.0      0.000000   7603.000000  12647.333333
fwd              11595.0   8249.000000   8396.387755   9811.800000
rwd              23949.6  24202.714286  14337.777778  21711.833333


body-style           wagon
drive-wheels
4wd            9095.750000
fwd            9997.333333
rwd           16994.222222
```

[26]:
```python
#Average price for body-style, heatmap to better understand distribution
fig, ax = plt.subplots()
im = ax.pcolor(grouped_pivot, cmap='RdBu')

#Label names
row_labels = grouped_pivot.columns.levels[1]
col_labels = grouped_pivot.index

#Move ticks and labels to the center
ax.set_xticks(np.arange(grouped_pivot.shape[1]) + 0.5, minor=False)
ax.set_yticks(np.arange(grouped_pivot.shape[0]) + 0.5, minor=False)

#Insert labels
ax.set_xticklabels(row_labels, minor=False)
ax.set_yticklabels(col_labels, minor=False)

#Rotate label if too long
plt.xticks(rotation=90)

fig.colorbar(im)
plt.show()
```



[28]:
```
#ANOVA for drive-wheels
grouped_test2=df_gptest[['drive-wheels', 'price']].groupby(['drive-wheels'])

f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'],␣
 ↪grouped_test2.get_group('rwd')['price'], grouped_test2.
 ↪get_group('4wd')['price'])
print( "ANOVA results: F =", f_val, "| P =", p_val)


#A large F-test score shows a strong correlation and a P-value of almost 0␣
 ↪implies almost certain statistical significance.
```

```
ANOVA results: F = 67.95406500780399 | P = 3.3945443577151245e-23
```

[29]:
```
#Check correlations of particular groups:

#ANOVA for drive-wheels / fwd and rwd
f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'],␣
 ↪grouped_test2.get_group('rwd')['price'])

print( "ANOVA results: F =", f_val, "| P =", p_val )
```

```
ANOVA results: F = 130.5533160959111 | P = 2.2355306355677845e-23
```

[30]:
```
#ANOVA for drive-wheels / 4wd and rwd
f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'],␣
 ↪grouped_test2.get_group('rwd')['price'])

print( "ANOVA results: F =", f_val, "| P =", p_val)
```

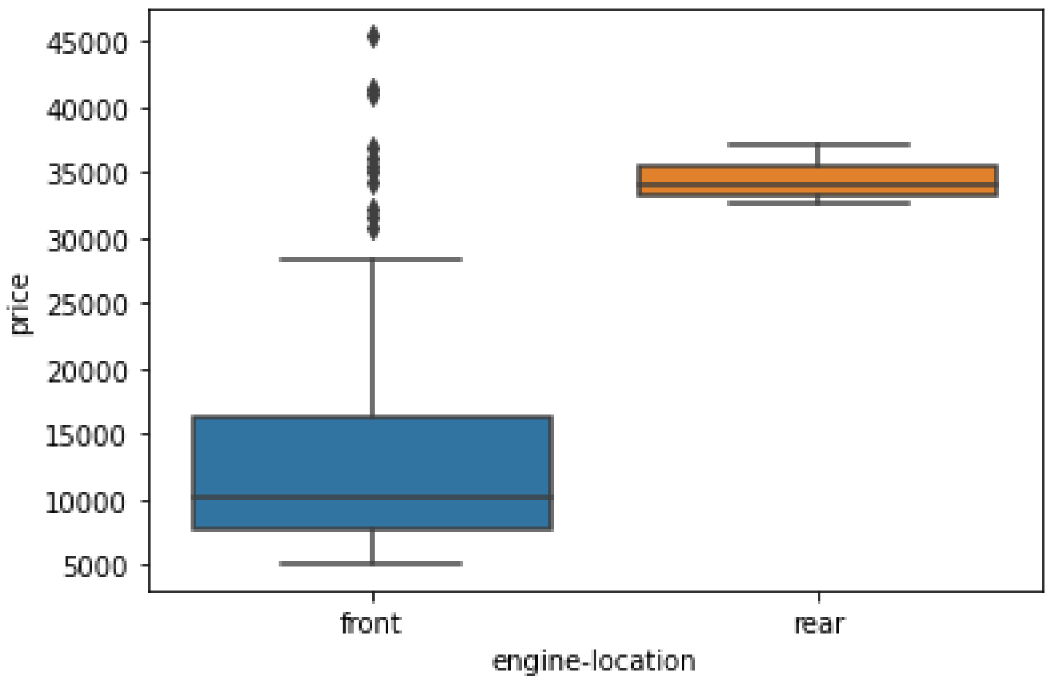
```
ANOVA results: F = 8.580681368924756 | P = 0.004411492211225333
```

[31]:
```
#ANOVA for drive-wheels / 4wd and fwd
f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'],␣
 ↪grouped_test2.get_group('fwd')['price'])

print("ANOVA results: F =", f_val, "| P =", p_val)
```

```
ANOVA results: F = 0.665465750252303 | P = 0.41620116697845666
```

[32]:
```
#Engine-location
sns.boxplot(x="engine-location", y="price", data=df)
```

[32]: <AxesSubplot:xlabel='engine-location', ylabel='price'>



[33]:
```
#Count per engine-location
engine_loc_counts = df['engine-location'].value_counts().to_frame()
engine_loc_counts.rename(columns={'engine-location': 'value_counts'},␣
 ↪inplace=True)
```

```
engine_loc_counts.index.name = 'engine-location'
engine_loc_counts

#exclude becasue there are only 3 rears
```

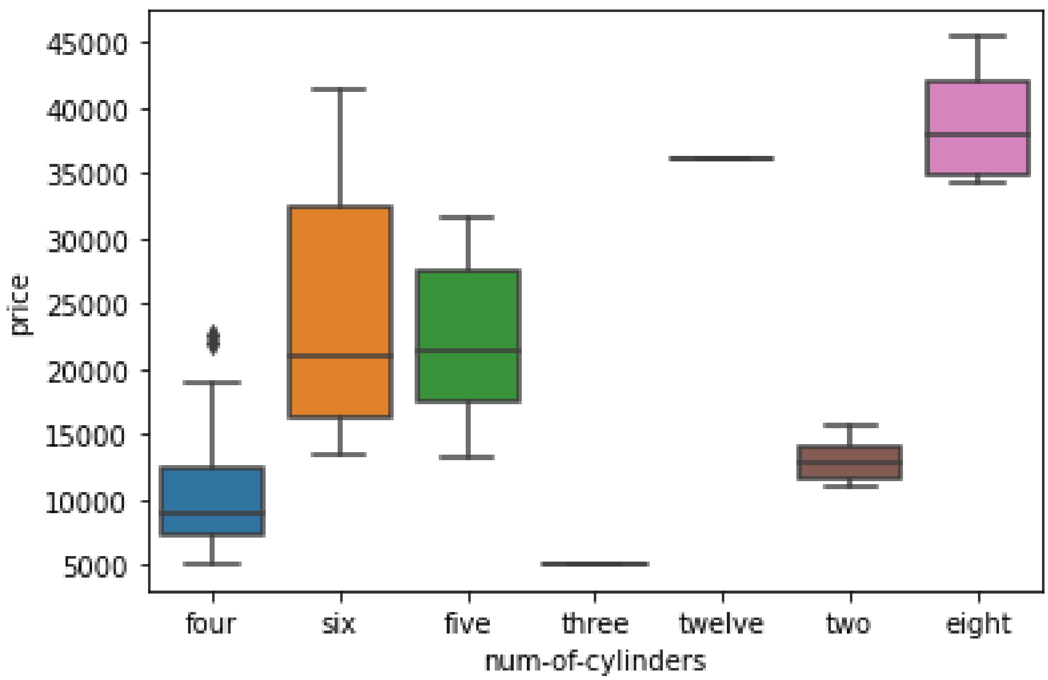
[33]:

```
                  value_counts
engine-location
front                      198
rear                         3
```

[34]:

```
#Num-of-cylinders
sns.boxplot(x="num-of-cylinders", y="price", data=df)
```

[34]: <AxesSubplot:xlabel='num-of-cylinders', ylabel='price'>



[35]:

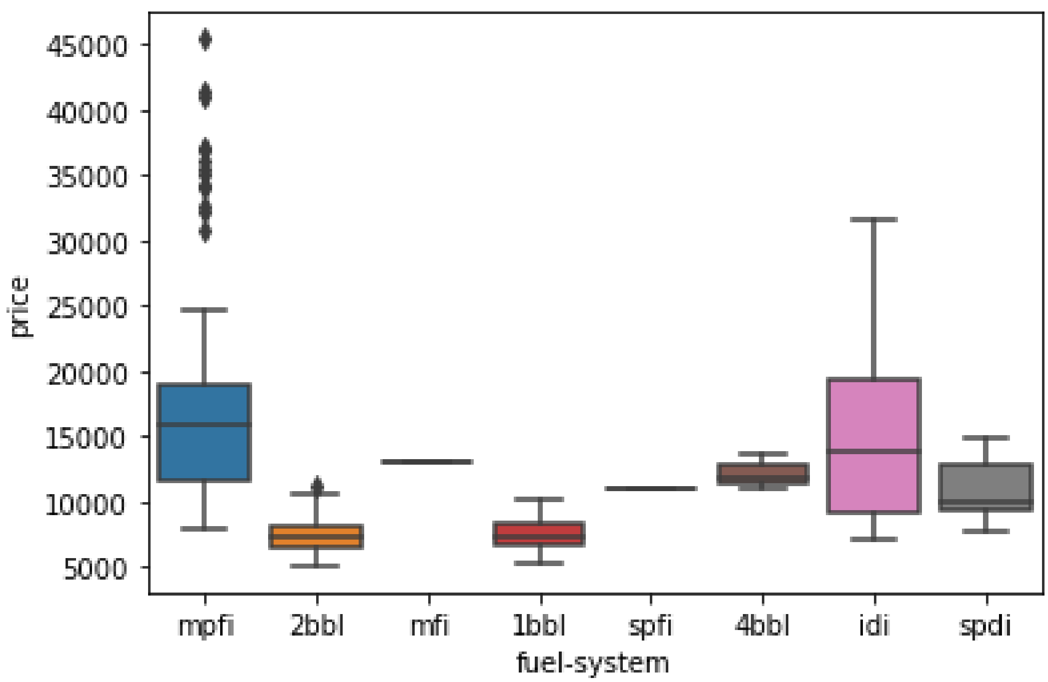
```
#Count per num-of-cylinders
engine_loc_counts = df['num-of-cylinders'].value_counts().to_frame()
engine_loc_counts.rename(columns={'num-of-cylinders': 'value_counts'},␣
 ↪inplace=True)
engine_loc_counts.index.name = 'num-of-cylinders'
engine_loc_counts

#exclude becasue most models fall into four cylinders category
```

[35]:

	value_counts
num-of-cylinders	
four	157
six	24
five	10
two	4
eight	4
three	1
twelve	1

[36]:
```
#Fuel-system
sns.boxplot(x="fuel-system", y="price", data=df)
#exclude due to overlapping
```

[36]: <AxesSubplot:xlabel='fuel-system', ylabel='price'>



Conclusion

[38]:
```
#The following should be considered as predictors:

#----Numerical----

#   Length
#   Width
#   Curb-weight
```

```
#   Engine-size
#   Horsepower
#   City-mpg
#   Highway-mpg
#   Wheel-base
#   Bore

#----Categorical----

#   Drive-wheels
```